Undecidability

1. What is undecidability?
2. The diagonalization method
3. Un(Countable) sets
4. Q is countable
5. R is uncountable
6. Some languages are not Turing-recognizable
7. ATM is undecidable
8. What is undecidability?

* A problem is undecidable if there is no algorithm that can solve it for all inputs (ex: software verification)

1. The diagonalization method

* Technique discovered by Cantor in 1873 that’s used to prove certain sets are larger than others.
* Cantor observed that two finite sets have the same size if the elements of one set can be paired with the elements of the other set. And this method can be extended to infinite sets.
* Assume we have sets A and B and a function f from A to B. f is one-to-one if it never maps different elements to the same place (f(a) != f(b)) whenever a != b. f is onto if it hits every elements of B (for every b from B there is an a from A and f(a) = b). A and B are the same size if there is a one-to-one, onto function f:A->B. A function that is both one-to-one and onto is called a correspondence. A correspondence is simply a way of pairing the elements of A with the elements of B.

1. Countable sets

* A set is countable if it’s either finite or can be put into a correspondence with N
* If an infinite set cannot be put into a correspondence with N then it is uncountable

1. Q is countable

* If we let Q = {m/n | m, n from N} be the positive set of rational numbers, Q seems to be much larger than N. Yet these two sets are the same size according to our definition. To demonstrate this we can list all the elements of Q, and then we pair the first element on the list with the number 1 from N, the second element with the number 2 from N, and so on. To get this list, we make an infinite matrix containing all the positive rational numbers as illustrated below so that the number i/j occurs in the *i*th row and *j*th column. Now we turn this matrix into a list using diagonalization. The first diagonal contains the single element 1/1, the second diagonal contains 2/1 and ½. So the first three elements of the list are 1/1, 2/1, ½. The third diagonal contains 3/1, 3/2, 3/3, but we need to skip 3/3 because it’s an element that would cause a repetition. So we only add 3/1, 3/2, and continuing in this way, we obtain a list of all the elements of Q.

1. R is uncountable

* Let R be the set of real numbers, to show that it is uncountable we show that no correspondence exists between N and R. The proof is by contradiction. Suppose that a correspondence f existed between N and R, for it to be a correspondence, f must pair all the members of N with all the members of R. But we will find an x in R that is not paired with anything in N, which will be our contradiction. The way we find this x is actually by constructing it. We choose each digit of x to make x different from one of the real numbers that is paired with an element of N. This can be illustrated through an example. Suppose that the correspondence f exists. Let f(1) = 3.14159…., f(2) = 55.555555…., f(3) = 1.1111…., and so on. The following table shows some values of a hypothetical correspondence f between N and R. We construct x, it is a number between 0 and 1 and any fractional digit will be different than the digits of the numbers already listed in this way. First digit we choose for x is anything but the first digit of the first number, second digit of x is anything but the second digit of the second number, and so on. Continuing in this way down the diagonal in the table for f, we obtain x, and we know that x is not f(n) for any n because it differs from f(n) in the *n*th fractional digit. This number is in R but is not in the list => R is uncountable (Note: we never select 0 or 9 when we construct x to stay away from a problem: 0.1999… and 0.2000…. are equal even though they have different representations).

1. Some Languages are not Turing-Recognizable

* Proof:

Let L be the set of all languages over alphabet Σ. We show that L is uncountable by giving a correspondence with B (the set of infinite binary sequences), thus showing the sets are the same size. Let Σ\* = {s1, s2, s3, ….}. Each language A from L has a unique sequence in B. The *i*th bit of that sequence is 1 if si is from A, 0 otherwise.

Example: if A were the language of all strings starting with 0 over the alphabet {0,1}, the sequence would be:

Σ\* = { ε, 0, 1, 00, 01, 10, 11, 000, 001, ….}

A = { , 0, 00, 01, 000, 001, ….}

Seq= 0 1 0 1 1 0 0 1 1 …..

f : L -> B, f(A) = the characteristic sequence of A is one-to-one and onto, hence is a correspondence. Therefore, as B is uncountable, L is uncountable as well.

* Conclusion: There are uncountable many languages but only countable TM so therefore some languages cannot be recognized by any TM.

1. ATM is undecidable

-> ATM = {⟨M,w⟩| M is a TM and M accepts w}

Proof by contradiction. We assume ATM is decidable. Let H be a decider for ATM where:

H(⟨M,w⟩) = accept if M accepts w

reject if M rejects w

We construct D using H as subroutine.

D(⟨M⟩):

1. Run H on ⟨M,⟨M⟩⟩

2. Output opposite of H (if H accepts, D rejects, if H rejects then D accepts)

In other words:

D(⟨M⟩) = accept if M doesn't accept ⟨M⟩

reject if M accepts ⟨M⟩

So when we try to run D with its own description ⟨D⟩ as input we get:

D(⟨D⟩) = accept if D doesn't accept ⟨D⟩

reject if D accepts ⟨D⟩

But this creates a paradox, D must do the opposite of what it does on its own input => contradiction => ATM is undecidable